IEM製品版用

本資料は、３Gシールドを簡単に利用するためのAruino用ライブラリ**a3gs**の解説書です。

IEM製品版3Gシールド ライブラリ仕様書(Ver1.01)

2012/12/01

著作：3Gシールドアライアンス



もくじ

Ⅰ編．IEM製品版３Gシールドの概要

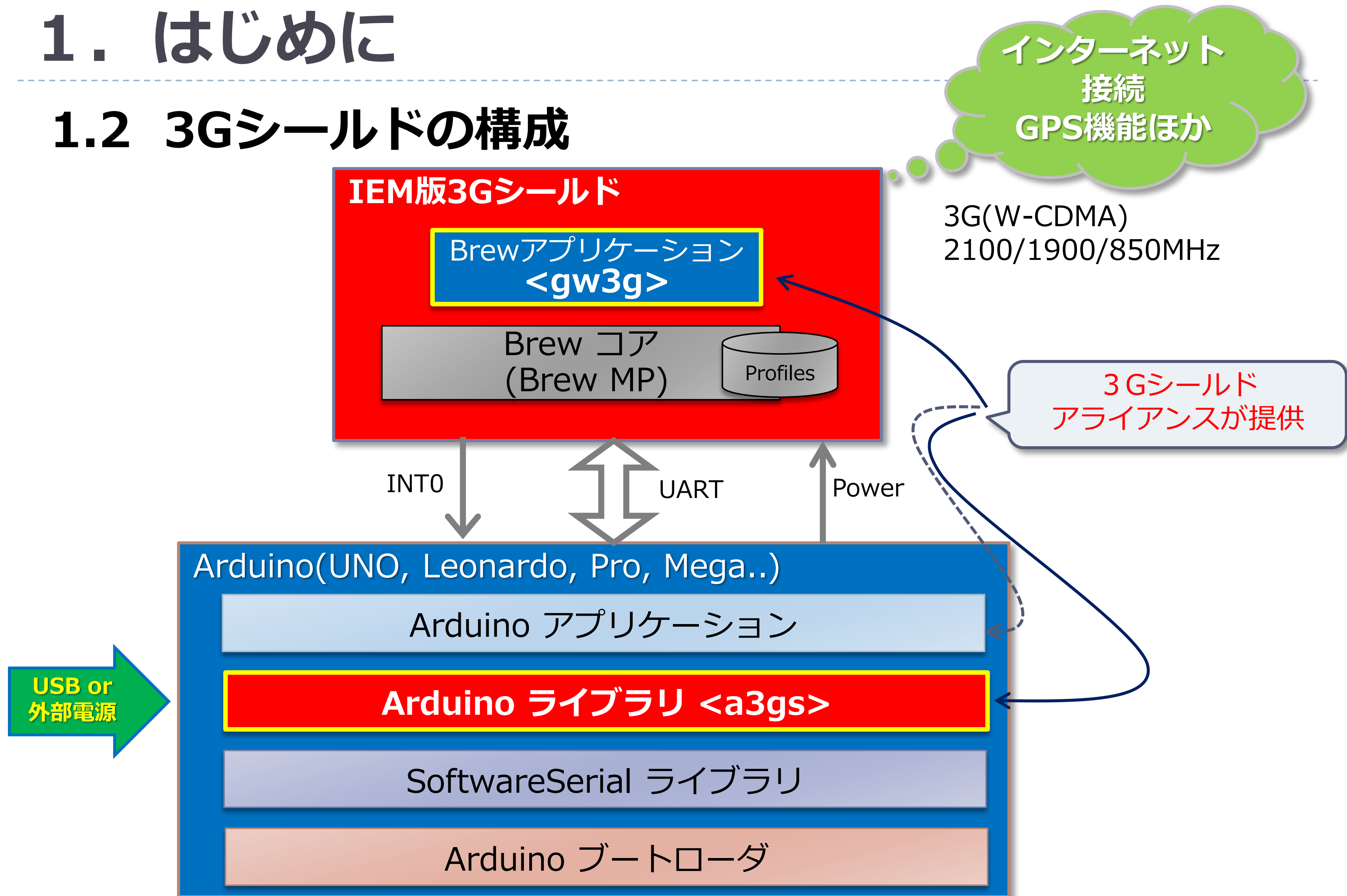
1．3Gシールドとは

- Arduino上で簡単に3G通信を行うことができるシールド
  - IEM製品版は、FOMA通信をサポート（NTTdocomo、MVNOのIIJmio、DTI等のSIMカードで利用可能）
（USB版は、イーモバイル系のUSBドングルによる通信をサポート予定）

- メモリサイズの小さい8ビットマイコン「Arduino」の上で利用できる高機能なライブラリ "a3gs" を提供

- ライブラリが提供する機能は、下記の通り：
  - **Web通信**(HTTP GET/POST)
  - **TCP/IP通信**(connect, disconnect, read, write)
  - **現在位置情報（GPS）取得**(GPS Standalone, AGPS)
  - **ショートメッセージ（SMS）送受信**(send, receive, check, onReceived)
  - **プロファイル取得・設定**(デフォルトのプロファイル(APN)の取得、設定)
  - **その他**（初期化、電源制御、時刻取得、電波強度取得、IMEI取得、LED制御等）

- 3Gシールドを利用するにあたっての留意点は下記の通り：
  - a3gsライブラリは、Arduino標準ライブラリSoftwareSerialを利用する。
  - INT0(割り込み0番)は、SMS着信通知で使用する。
  - 3Gシールドで使用するピンについては、「取扱説明書」を参照のこと。



TABrain a3gsa

Ⅱ編　３Gシールドライブラリ概説

【注意】関数名の右上に※印があるものは、Arduino GSM/GPRS シールド用ライブラリと互換性がある関数

1．IEM(Internet of Everything Module)とは

- 小型の3G通信モジュールの特徴
  - 韓国AnyDATA社の「DTW400-W」（JATE/TELEC 取得済）を利用
  - Qualcomm社のチップセット「QSC6240」を採用
  - サイズは 21mm × 22mm × 4.5mm，重量は4.5ｇ と非常に小型
  - 携帯向けに設計されたモジュールであり、消費電力が低い



DTW400-W裏表と100円玉

DTW400-Wの主な仕様	
UMTS	850/1900/2100 MHz
EDGE/GPRS/GSM	850/900/1800/1900 MHz
GPS	Standalone GPS, AGPS
Speed	384Kbps(Download)/384Kbps(Upload)
OS	Brew MP 1.0.4
その他	JATE　取得済み
動作温度	-20℃ ～ 60℃

2．ライブラリ"a3gs"の概説
3Gシールドの制御ライブラリ

分類	メソッド名※1	機能概要	補足
コントロール（Control）	getStatus※	3Gシールドの状態取得	
	begin※	ライブラリの初期化	
	end※	ライブラリの終了	
	restart※	3Gシールドのリセット	
	start※	3Gシールドの電源ON	
	shutdown※	3Gシールドの電源OFF	
	getIMEI	IMEI-IDの取得	
	setLED	LED1のON/OFF	
	setBaudrate	UARTの通信速度の設定	初期は4800bps
ショートメッセージ（SMS） *	sendSMS※	SMSの送信	
	availableSMS※	SMSの受信状態チェック	
	readSMS※	SMSの読出し	
	onSMSReceived	SMS着信時のコールバック設定	

※　Arduino GSM/GPRSシールド用ライブラリと互換性がある関数です。
*　利用するSIMカードによって使えない場合があります。

2．ライブラリ"a3gs"の概説

3Gシールドの制御ライブラリ

分類	メソッド名	機能概要	補足
Web機能	httpGET※	GETメソッドの要求	
	httpPOST	POSTメソッドの要求	
	tweet※	Twitterへの投稿	*
現在位置取得（GSP)	getLocation	現在位置の取得	内蔵GPSを使用
通信機能その他	getServices	利用可能サービスの取得	
	getRSSI	電波強度の取得	
	getTime	現在時刻の取得	日付・時刻形式
	getTime2	現在時刻の取得	通算秒形式
	getVersion	3Gシールドのバージョンの取得	

※　Arduino GSM/GPRSシールド用ライブラリと互換性がある関数
*　無償サービス「http://arduino-tweet.appspot.com/」を利用（要登録）

2．ライブラリ"a3gs"の概説

3Gシールドの制御ライブラリ

分類	メソッド名	機能概要	補足
TCP/IP機能	connectTCP※	TCPコネクションを接続	
	disconnectTCP※	TCPコネクションを切断	
	read※	データの読込み	2つのﾊﾞﾘｴｰｼｮﾝ有
	write※	データの書出し	3つのﾊﾞﾘｴｰｼｮﾝ有
プロファイル	setDefaultProfile	デフォルトプロファイルを設定	
	getDefaultProfile	デフォルトプロファイルを取得	

※　Arduino GSM/GPRSシールド用ライブラリと互換性がある関数

2．ライブラリ"a3gs"の概説

3Gシールドの制御ライブラリで使用する定数

- 各ライブラリを利用する上で、留意すべき定数（主に最大値の定義）を下表に示す。これらは、ヘッダファイル"a3gs.h"で定義されている。

分類	定数名	意味	補足
SMS	a3gsMAX_SMS_LENGTH	SMSメッセージの最大バイト数	
	a3gsMAX_MSN_LENGTH	電話番号の最大バイト数(最大桁数)	
Web	a3gsMAX_URL_LENGTH	URLの最大バイト数	※1
	a3gsMAX_HEADER_LENGTH	POSTのヘッダの最大バイト数	※1
	a3gsMAX_BODY_LENGTH	POSTのボディの最大バイト数	※1
	a3gsMAX_RESULT_LENGTH	GET/POSTのレスポンスの取得可能な最大バイト数	※1
	a3gsMAX_TWEET_LENGTH	ツイートメッセージの最大バイト数	※1
TCP/IP	a3gsMAX_HOST_LENGTH	ホスト名の最大バイト数	※1
	a3gsMAX_DATA_LENGTH	一度に読み書きできるデータの最大バイト数	※2
Misc.	a3gsMAX_PROFILE_NUMBER	プロファイル数	

※1 これらの定数は、ATmega328*/32U*（Unoなど）を利用したArduinoではSRAMのサイズが小さいことからかなり制約が厳しい。ATmega2560/1280（Megaなど）またはADKを利用することで、これらの最大値を大きくすることができる。
※2 大きなデータを読み書きする場合は、複数回に分けてread/writeを実行する。

3．ライブラリの使用方法（例）

- 本ライブラリは、スケッチの中で以下の順序でコントロール用のメソッド（関数）を呼び出すことにより、利用できるようになる：

```
#include "a3gs.h"

void setup()
{
    . . .
    if (a3gs.start() == 0) {
        // 3Gシールドの電源ONに失敗した
    }
    if (a3gs.begin() == 0) {
        // ライブラリの初期化に失敗した
    }
       // 3Gシールドが使用できるようになった
    . . .
}
```

a3gs.h を宣言

a3gsの状態を検査

TABrain a3gsa

Ⅲ編　３Gシールドライブラリ

本稿では、a3gsライブラリR1.0の使用について説明する。

【注意】関数名の右上に※印があるものは、Arduino GSM/GPRS シールド用ライブラリと互換性がある関数

1. コントロール関連の関数

提供関数ライブラリ

分類	メソッド名※1	機能概要	補足
コントロール（Control）	getStatus※	3Gシールドの状態取得	
	begin※	ライブラリの初期化	
	end※	ライブラリの終了	
	restart※	3Gシールドのリセット	
	start※	3Gシールドの電源ON	
	shutdown※	3Gシールドの電源OFF	
	getIMEI	IMEIの取得	
	setLED	LED1のON/OFF	
	setBaudrate	UARTの通信速度の設定	次回のリセット後に有効となる

※　Arduino GSM/GPRSシールド用ライブラリと互換性がある関数

概要

- IEMライブラリの初期化・終了、ライブラリの状態の取得、3Gシールドのリセット、電源ON/OFF、IMEIの取得、LED1のON/OFF、UARTの通信速度の設定を行う

留意点

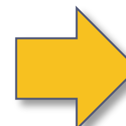
- 電源ONならびにリセットには、40秒程度の時間が掛かる
- 電源OFFには、15秒程度の時間が掛かる
- setBaudrateによる通信速度の変更には、十分留意すること

1. コントロール関連の関数　getStatus※

int getStatus(void)	
機能概要	現在のライブラリの状態を取得
引数	なし
戻り値	現在の状態（ERROR, IDLE, READY, TCPCONNECTEDCLIENT のいずれか）
補足	各状態の意味は下記の通り： A3GS::ERROR：エラーが発生した A3GS::IDLE：空いている（機能の実行が可能） A3GS::READY：同上 A3GS::TCPCONNECTEDCLIENT：TCPコネクションが接続した

【利用例】

```
int  status;
status = a3gs.getStatus();
Serial.print("Status is ");
switch (status) {
  case A3GS::ERROR :      Serial.println("ERROR");    break;
  case A3GS::IDLE :        Serial.println("IDLE");       break;
  case A3GS::READY :      Serial.println ("READY");     break;
  case A3GS::TCPCONNECTEDCLIENT :
              Serial.println("TCPCONNECTEDCLIENT");     break;
  default : Serial.println("Unknown");    break;
```



【出力結果例】

```
Status is IDLE
```

1. コントロール関連の関数　begin※

● バリエーション1

int begin(char* pin)	
機能概要	ライブラリを初期化する
引数	**pin**：未使用(指定は不要)
戻り値	**0**：正常に初期化を実行できた時
	1：エラーが発生した時（ライブラリは使用不可）
	2：IEM上のgw3gアプリのバージョンが古い（ライブラリは使用不可）
補足	3Gシールドの電源がONの状態で、本ライブラリの使用に先立って一度だけ本関数を呼び出す必要がある。 初期化に失敗した場合は、end()関数を呼び出して、時間をおいて何度かリトライすることで成功する場合がある。 終了関数end()を呼び出した後は、再度、本関数を呼び出すことができる。 本関数の中では、デフォルトの通信速度で標準ライブラリ「SoftwareSerial」を初期化(begin)する。

【使い方の例】

```
if (a3gs.begin() == 0)
  Serial.println("Succeeded.");
```

1. コントロール関連の関数　begin※

● バリエーション2

int begin(char* pin, uint32_t baudrate)	
機能概要	ライブラリを初期化する
引数	**pin :** 未使用(指定は不要)
	baudrate : 設定する通信速度 (1200/2400/4800/9600/19200/38400/57600/115200)
戻り値	**0**：正常に初期化を実行できた時
	1：エラーが発生した時（ライブラリは使用不可）
	2：IEM上のgw3gアプリのバージョンが古い（ライブラリは使用不可）
補足	3Gシールドの電源がONの状態で、本ライブラリの使用に先立って一度だけ本関数を呼び出す必要がある。 初期化に失敗した場合は、end()関数を呼び出して、時間をおいて何度かリトライすることで成功する場合がある。 終了関数end()を呼び出した後は、再度、本関数を呼び出すことができる。 本関数の中では、指定された通信速度で標準ライブラリ「SoftwareSerial」を初期化(begin)する。 通信速度の変更は、setBaudrate()関数を使って事前に行っておく必要がある。

【使い方の例】

```
if (a3gs.begin(0, 9600) == 0)
  Serial.println("Succeeded.");
```

1．コントロール関連の関数　end※

int end(void)	
機能概要	ライブラリの使用を終了する
引数	なし
戻り値	**0**：正常に終了処理を実行できた時
	0以外：エラーが発生した時
補足	本関数の中では、標準ライブラリであるSoftwareSerialを終了(end)する。

【使い方の例】
・次頁参照

1. コントロール関連の関数　restart※

int restart(char* pin)	
機能概要	3Gシールドを再起動（リセット）する
引数	**pin**：未使用(指定は不要)
戻り値	**0**：正常にリセットを実行できた時
	0以外：エラーが発生した時（リセットできない時）
補足	IEM全体をリセットする。 通常、本関数を呼び出してから10秒程度でIEMはリセット処理を開始し、40秒程度で利用可能な状態となる。 本関数によるリセット後に再度ライブラリを利用する場合は、一旦、終了関数end()を呼び出した後に、初期化関数begin()を呼び出すこと。

【使い方の例】

```
if (a3gs.restart() == 0) {
    Serial.println("Restarting..");
    a3gs.end();
    if (a3gs.begin() == 0)
        Serial.println("I'm OK.");
}
else
    Serial.println("Restart Failed.");
```

1. コントロール関連の関数　start※

int start(char* pin)	
機能概要	3Gシールドの電源をONにする
引数	**pin**：未使用(指定は不要)
戻り値	**0**：正常に電源ONを実行できた時
	0以外：エラーが発生した時（電源ONできない時）
補足	本関数を呼び出しには、40秒程度掛かる（本関数の呼び出しが完了した時点で、3Gシールドが利用可能な状態となっている）。その後、初期化関数begin()を呼び出すことで本ライブラリを利用することができる。

【使い方の例】

```
if (a3gs.start() == 0 && a3gs.begin()) {
   Serial.println("Succeeded.");
      // 成功処理
}
else
    Serial.println("Restart Failed.");
```

1. コントロール関連の関数　shutdown※

int shutdown(void)	
機能概要	3Gシールドの電源をOFFにする
引数	なし
戻り値	**0**：正常に電源OFFを実行できた時
	0以外：エラーが発生した時（電源OFFできない時）
補足	本関数を呼び出しには、15秒程度掛かる 本関数を呼び出した後は、再度、電源ON関数start()を呼び出すことで3Gシールドを利用することができる。

【使い方の例】

```
a3gs.end();
a3gs.shutdown();
```

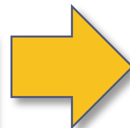

TABrain a3gsa

1. コントロール関連の関数　getIMEI

int getIMEI(char* imei)	
機能概要	3Gシールドに装着されているIEMのIMEIを取得する
引数	**imei** ：取得したIMEI（サイズは **a3gsIMEI_SIZE** バイト）
戻り値	**0**：正常に取得できた時
	0以外：エラーが発生した時（取得できない時）
補足	IMEIとは３G通信モジュール(IEM)の識別IDである（電話番号とは無関係） 引数imeiが指す結果格納場所のスペース（**a3gsIMEI_SIZE**バイト＝16桁）は、あらかじめ呼び出し側で確保しておくこと。

【使い方の例】

```
char imei[a3gsIMEI_SIZE];
if (a3gs.begin() == 0)  {
   a3gs.getIEI(imei);
   Serial.println(imei);
}
```



【出力結果例】

```
354563020267950
```

このIMEIの中に、IEMモジュールに記載されたIDが含まれています。

TABrain a3gsa

1. コントロール関連の関数　setLED

int setLED(boolean sw)	
機能概要	3Gシールドに搭載されているLED1を制御する
引数	**sw** **：**ONにする時はTRUE、OFFにする時はFALSEを指定する
戻り値	**0**：正常に設定できた時
	0以外：エラーが発生した時
補足	LED1(緑色のLED)が3Gシールドのどこの位置に配置されているか等は、「取扱説明書」を参照のこと。

【使い方の例】

```
if (aFlag) {
  a3gs.setLED(TRUE);
  led1_status = TRUE;
  Serial.println("LED1 is turned on.");
}
```

1. コントロール関連の関数　setBaudrate

int setBaudrate(int baudrate)	
機能概要	Arduinoと3Gシールドを仲介するUARTの通信速度を設定する
引数	**baudrate** ：設定する通信速度(2400/4800/9600/19200/38400/57600のいずれか)
戻り値	**0**：正常に変更できた時
	0以外：エラーが発生した時
補足	**本関数の利用には十分留意すること。不適切な値を設定した場合は、3Gシールドを利用することができなくなる可能性がある。** 工場出荷時の通信速度は、安定動作が可能な 4800(bps) となっている。 通信速度をデフォルトの設定値よりも高くするには、ハードウェアシリアルの利用を推奨する。 本関数による通信速度の変更は、次回のIEMのリセット後で有効となる。リセットするまでは、変更前の通信速度が維持される。 デフォルトの通信速度と異なる通信速度を設定した場合は、次回の初期化の際にはbegin()を呼び出すこと（詳細はbegin()の項を参照）

【使い方の例】

```
if (a3gs.setBaudrate(9600) == 0)  {
  Serial.println("Baudrate was changed.");
  Serial.println("Please reset me now.");
}
```

2．ショートメッセージ関連の関数

提供関数ライブラリ

分類	メソッド名※1	機能概要	補足
ショートメッセージ（SMS）	sendSMS※	SMSの送信	
	availableSMS※	SMSの受信状態チェック	
	readSMS※	SMSの読出し	
	onSMSReceived	SMS着信時に呼び出す関数を設定	

概要

- 3GのSMS（Short Message Service：100文字程度までの簡易メッセージングサービス）を利用する
- SMSは、通信キャリアにまたがって送受信できる

留意すべき点

- 使用するSIMカード（通信サービス）により、SMSが利用できない場合がある
- 通信料金（送信側に課金、一般には定額プランの範囲外）に注意すること
- 通信回線の状態によっては、SMSの配送遅延が起こる場合がある（常に即時配送できるとは限らない）
- SMSが届かない場合は、受信側のSMS受信拒否設定にも注意すること
- SMSのメッセージの中に利用できない文字が存在することに注意すること（例えば、"@"文字等は使えない）

2．ショートメッセージ関連の関数　sendSMS※

int sendSMS (const char* to, const char* msg, int encode)	
機能概要	SMS（ショートメッセージ）を指定した宛先へ送信する
引数	**to**：送付先の電話番号（ハイフォンなしの10ケタの半角数字）
	msg：送信するメッセージ（最大**a3gsMAX_SMS_LENGTH**文字）
	encode：メッセージに使用するエンコード方法（**a3gsCS_***で指定）。省略可 （省略時はASCIIエンコードと解釈）
戻り値	**0**：正常に送信できた時
	0以外：送信できなかった時
補足	本機能を利用するためには、SMSが利用できる通信サービス(SIMカード)を利用する必要がある。 関数getServices()にて、**a3gsSRV_CS**または**a3gsSRV_BOTH**のいずれかの返却値が取得できる場合にはSMSは利用できる。 日本語は**UNICODE**でのみ取り扱いが可能である。Arduinoの場合、デフォルトの文字コードは**UTF-8**であり、UNICODEは16進数等で直接表現する必要がある。

【使い方の例】

```
if (a3gs.setBaudrate(9600) == 0) {
  Serial.println("Baudrate was changed.");
  Serial.println("Please reset me now.");
}
```

2．ショートメッセージ関連の関数　availableSMS※

boolean availableSMS (void)	
機能概要	SMSが届いているかをチェックする
引数	なし
戻り値	**true**：SMSが届いている時
	false：SMSが届いていない時
補足	現在の3Gシールドのバージョンでは、SMSは1度に1通だけ受信・蓄積できる。2通以上が届く場合は、最新以外のSMSは捨てられる。

【使い方の例】

```
char msg[a3gsMAX_SMS_LENGTH+1], msn[a3gsMAX_MSN_SIZE+1];
void loop() {
  if (a3gs.availableSMS())  {
    int msgLen = sizeof(msg);
    int msnLen = sizeof(msn);
    a3gs.readSMS(msg, msgLen, msn, msnLen);
    //  受信したSMSの処理
  }
  delay(1000);
}
```

2．ショートメッセージ関連の関数　readSMS※

boolean readSMS (char* msg, int msglength, char* number, int nlength)	
機能概要	受信したSMSを読み出す
引数	**msg**：[OUT] 読み出したメッセージ（最大**a3gsMAX_SMS_LENGTH**文字）
	msglength：msgのサイズ（バイト数）
	number：[OUT] SMSの送信元の電話番号
	nlength：numberのサイズ（バイト数） （通常、**a3gsMAX_MSN_LENGTH**文字を確保）
戻り値	**true**：正常に読み出した時
	false：読み出し不可の時（SMSを未受信、SMS使用不可 等）
補足	正常に読み出せた場合は、msgおよびnumberは'¥0'文字で終端される。 メッセージはASCIIまたはUNICODEで読み出すことができる。 上記以外は、sendSMS()を参照

2．ショートメッセージ関連の関数　onSMSReceived

int onSMSReceived (void (*handler)(void))	
機能概要	SMSが着信した時に呼び出される関数を設定する
引数	**handler**：呼び出される関数（戻り値・引数は無）へのポインタ
戻り値	**0**：正常に設定できた時
	0以外：設定できなかった時
補足	本機能は、ArduinoのD3ピンの割込み(INT0/LOW)を利用する。 handler関数は、割り込み処理として呼び出されるため、割り込み中に利用できない機能に注意すること。また、長時間かかる処理は避けること（handler関数の実行中は、3Gシールドからの受信データは取りこぼす可能性がある）

【使い方の例】

```
void ledOn(void) {
  digitalWrite(LED_PIN, HIGH);
  ledStatus = HIGH;
}

void setup() {
  if (a3gs.start() == 0 && a3gs.begin() =- 0)  {
    a3gs.onSMSReceived(ledOn);
  }
  // . . .
}
```

3．Web関連の関数

提供関数ライブラリ

分類	メソッド名	機能概要	補足
Web機能	httpGET※	GETメソッドの要求	
	httpPOST	POSTメソッドの要求	
	tweet※	Twitterへの投稿	*

※　Arduino GSM/GPRSシールド用ライブラリと互換性がある関数
*　無償サービス「http://arduino-tweet.appspot.com/」を利用（要Twitterの登録）

概要

- httpを簡単に利用できる。
- GET/POSTメソッドを利用できる。
- Web機能の関数は、すべて同期処理である。そのため、レスポンスが取得できるまで、あるいは通信がタイムアウト(30秒程度)するまで呼び出し元には制御は戻らない。
- tweetは、サードパーティのフリーサービスを利用することで使用できる（ユーザ登録が必要、利用条件はそのサービスに従う）。
  - 詳細は http://arduino-tweet.appspot.com/ （3Gシールドアライアンスとは関係のないサービス）

留意点

- 使用するSIMカードで、3Gパケット通信が利用できること
- 通信料金（http通信の利用は、通常、定額プランの範囲内）に留意すること

3．Web関連の関数　httpGET ※

int httpGET (const char* server, uint16_t port, const char* path, char* result, int resultlength)	
機能概要	指定したサーバ、ポート、パスに対して、http/GETリクエストを発行して、そのレスポンスを返却する
引数	**server**：サーバのドメイン名
	port：サーバのポート番号（通常は80を指定）
	path：URLのパス
	result：[OUT] レスポンスの格納先（スペースは呼び出し側で確保）
	resultlength：resultのサイズを指定
戻り値	**0**：正常にGETできた時
	0以外：GETできなかった時
補足	引数serverに、IPアドレスを直接指定することはできない。 本関数の実行時間は、状況によって1～20秒程度掛かる。 サーバからのレスポンスのサイズが大きい場合は、resultlength以降のデータは破棄される。 resultは、'¥0'文字で終端される。文字コードは、接続先のサーバに依存する。 また、レスポンスにはヘッダは含まれない（ボディ部分のみ）

3．Web関連の関数　httpPOST

int httpPOST (const char* server, uint16_t port, const char* path, const char* header, const char* body, char* result, int* resultlength)	
機能概要	指定したサーバ、ポート、パスに対して、http/POSTリクエストを発行して、そのレスポンスを返却する
引数	**server**：サーバのドメイン名
	port：サーバのポート番号（通常は80を指定）
	path：URLのパス
	header：HTTPのヘッダ文字列(最大**a3gsMAX_HEADER_LENGTH**バイト)
	body：HTTPのボディ文字列(最大**a3gsMAX_BODY_LENGTH**バイト)
	result：[OUT] レスポンスの格納先（スペースは呼び出し側で確保）
	resultlength：[IN/OUT] resultのサイズを指定、呼び出し結果のサイズが返却
戻り値	**0**：正常にPOSTできた時
	0以外：POSTできなかった時

次ページへ続く

3．Web関連の関数　httpPOST

前ページ
から続く

int httpPOST (const char* server, uint16_t port, const char* path, const char* header, const char* body, char* result, int* resultlength)	
補足	引数headerでは、Content-Lengthの指定は不要である。 引数bodyでは、最後の空行は指定不要である。 引数serverに、IPアドレスを直接指定することはできない。 本関数の実行時間は、状況によって1～20秒程度掛かる。 サーバからのレスポンスのサイズが大きい場合は、resultlength以降のデータは破棄される。 resultは'¥0'文字で終端される。文字コードは、サーバの処理に依存する。 レスポンスには、ヘッダは含まれない（ボディ部分のみ） 引数headerおよびbodyでは、下記の"$"文字を使ったエスケープシーケンスをサポートする（直接、制御文字を指定することはできない）： 　$t : TAB(0x09) 　$r : CR(0x0d) 　$n : NL(0x0a) 　$" : "そのもの 　$$: $そのもの 　$xhh または $Xhh : 16進数hh(スケッチにおける"0xhh"と同義)

3．Web関連の関数　tweet ※

int tweet (const char* token, const char* msg)	
機能概要	Twitterへ投稿する
引数	**token**：アクセスに必要なトークン（認証情報） **msg**：投稿するメッセージ（最大**a3gsMAX_TWEET_LENGTH**バイト）
戻り値	**0**：正常に投稿できた時 **0以外**：投稿できなかった時
補足	tweetは、下記のフリーサービスを利用することで使用できる。ユーザ登録が必要で、利用条件はこのサービスに従う。 詳細は http://arduino-tweet.appspot.com/ を参照のこと。 【注意】上記サービスの制限により、同一メッセージを連続して投稿することはできない。また、一定時間内に投稿できるメッセージ数に制限がある。この制限を守らない場合は、正しく引数を指定している場合でも本関数はエラーを返却する。

4．現在位置取得（GPS）関連の関数

提供関数ライブラリ

分類	メソッド名	機能概要	補足
現在位置取得（GPS)機能	getLocation	現在位置の取得	内蔵GPSを使用

概要

- IEM内蔵GPSや3Gネットワークを利用して位置を測位
- 引数の指定により、下記のいずれかの測位方法を選択できる：
  - **a3gsMPBASED**
    - GPSを利用して現在位置を測位する。GPSが利用できない場合は、3Gネットワークを利用する。
  - **a3gsMPASSISTED**
    - 3Gネットワーク上のロケーションサーバを利用して現在位置を測位する。
  - **a3gsMPSTANDALONE**
    - GPSのみを利用して現在位置を測位する。

留意点

- 通信サービス（例えば、IIJmio等）によっては、3Gネットワーク上のロケーションサーバを利用することができない。その場合は、GPS単独の測位のみが利用できる。
- 測位方法としてa3gsMPSTANDALONEを指定した場合は、通信料金（通常、定額プランの範囲内だが、SIMカードの通信サービスによる）が発生する
- 屋内や都心等のように、上空にある衛星の電波がGPSアンテナで補足できない場所では、正しく測位できない場合がある。

4．現在位置取得（GPS）関連の関数　getLocation

int getLocation(int method，char* latitude，char* longitude)	
機能概要	現在位置を取得する
引数	**method**：測位方法（**a3gsMPBASED** / **a3gsMPASSISTED** / **a3gsMPSTANDALONE** のいずれか）を指定
	latitude：[OUT] 緯度(北緯) 9.99999形式、ただし桁数は場合により可変
	longitude：[OUT] 経度(東経) 同上
戻り値	**0**：正常に取得できた時
	0以外：取得できなかった時（latitude, longitudeの値は不定）
補足	本関数の実行には、数十秒～3分程度の時間が掛かる。 AGPSサーバは、３Gシールド(Ver1.0)ではGoogleのサーバを利用する（今後変更となる可能性がある） 本関数は、同期処理である。そのため、測位処理が完了するまで、あるいは測位が失敗するまで呼び出し元には制御は戻らない。

5．通信その他機能の関数

- **提供関数ライブラリ**

分類	メソッド名	機能概要	補足
通信その他機能	getServices	利用可能サービスの取得	
	getRSSI	電波強度の取得	
	getTime	現在時刻の取得	日付・時刻形式
	getTime2	現在時刻の取得	通算秒形式
	getVersion	3Gシールドのバージョンの取得	

- **留意点**
  - 時刻は、使用している3Gネットワークから取得する（タイムゾーンや時刻精度は利用するネットワークに依存する）

5．通信その他機能の関数　getServices

int getServices(int& status)	
機能概要	現在利用できるネットワークサービスを取得する
引数	**status**：[OUT] 利用できるネットワークサービス（下記のいずれか） **a3gsSRV_NO**（＝０）：サービス利用不可 **a3gsSRV_PS**（＝１）：パケット通信サービスのみ **a3gsSRV_CS**（＝２）：音声通信(+SMS)サービスのみ **a3gsSRV_BOTH**（＝３）：パケット通信、音声通信(SMS)いずれも可
戻り値	**0**：正常に取得できた時
	0以外：取得できなかった時（statusの値は不定）

【使い方事例】

```
int status;
if (a3gs.getServices(status) == 0)
    Serial.println(status);
```

SIMカードによる提供サービスの違いを取得

5．通信その他機能の関数　getRSSI

int getRSSI(int& rssi)	
機能概要	3Gの電波強度（RSSI）を取得する
引数	**rssi**：[OUT] 取得した電波強度（単位はdBm）　<範囲：-1 ～ -128>
戻り値	**0**：正常に取得できた時
	0以外：取得できなかった時（rssiの値は不定）
補足	電波強度は必ずマイナス値が返却される。0に近いほど電波強度は強い。

【電波受信レベルの測定サンプル】
　3Gシールドアライアンス側で測定した場合、３Gアンテナを付けないで測定した場合、「-125dBm」を計測。
また感度の良いところでは、「-68dBm」を計測。

【使い方事例】

```
int rssi;
if (a3gs.getRSSI(rssi) == 0)
    Serial.println(rssi);
```

5．通信その他機能の関数　getTIME

int getTime(char* date, char* time)	
機能概要	現在の日付・時刻を取得する
引数	**date**：[OUT] 取得した日付（"YYYY-MM-DD"形式）
	time：[OUT] 取得した時刻（"HH:MM:SS"形式）
戻り値	**0**：正常に取得できた時
	0以外：取得できなかった時（date/timeの値は不定）
補足	日付・時刻は使用している3Gネットワークから取得するため、精度およびタイムゾーンはネットワークに依存する。（日本国内で利用する場合は、タイムゾーンは日本(JST)となる） 時刻は24h制（固定）である。ただし、自動調整出力となるため、変更は不可。

【使い方事例】

【利用例】

```
if (a3gs.getTime(date, time) == 0) {
    Serial.print(date);
    Serial.print(" ");
    Serial.println(time);
}
```

【出力結果例】

```
2012/12/01　12:10:43
```

【注意：時刻自動調整機能について】
長く電源を入れていないと、時刻が一旦1980年1月5日16:00:00に戻る。
電源を入れて動かしている間に、一時的に現在時刻より16時間遅れで表示され、
さらに、日本時間で表示されるようになる。

5．通信その他機能の関数　getTIME2

int getTime2(uint32_t& seconds)	
機能概要	現在の時刻（1970/1/1からの通算秒：「UNIX時間」とも言う）を取得する
引数	**seconds**：[OUT] 取得した通算秒
戻り値	**0**：正常に取得できた時
	0以外：取得できなかった時（date/timeの値は不定）
補足	時刻を秒単位で取得できるため、時刻の比較処理等が簡単となる。 日付・時刻の精度およびタイムゾーンについては、getTime()を参照。 秒への換算処理では、閏（うるう）年も考慮している。 2038年※問題が発生する可能性がある。

※ **2038年問題**は、ISOの通算秒の定義に1970年1月1日からとしていて、C言語の標準で32ビット符号付intを採用している場合、2038年1月19日3時14分7秒（UTC、以下同様）を過ぎると、この値がオーバーフローし、負と扱われるため、コンピュータが誤動作する可能性があるとされる問題。
【詳細はウィキペディア参照】

5．通信その他機能の関数　getVersion

int getVersion(char* version)	
機能概要	3Gシールド(IEM上のgw3gアプリ)のバージョンを取得する
引数	**version**：[OUT] 取得したバージョン("9.9"形式)
戻り値	**0**：正常に取得できた時
	0以外：取得できなかった時（versionの値は不定）
補足	begin()の処理の中で3Gシールドのバージョンと本ライブラリの整合性をチェックしているため、通常、本関数を利用する必要はない。

6．TCP/IPの関数

提供関数ライブラリ

分類	メソッド名	機能概要	補足
TCP/IP	connectTCP※	TCPコネクションを接続	
	disconnectTCP※	TCPコネクションを切断	
	read※	データの読込み	2バリエーション有
	write※	データの書出し	3バリエーション有

※ Arduino GSM/GPRSシールド用ライブラリと互換性がある関数

【注意事項】

- TCP/IP v4のみサポートする。
- 一度に一つのコネクションだけを利用できる。（Web機能とは独立して使用できる）
- 本機能で提供する関数では、すべて同期的に処理する。そのため、サーバから結果が得られるまで、エラーが発生するまで、あるいは通信がタイムアウトするまで呼び出し元には制御が戻らない。
- 接続や通信では、タイムアウト時間として30秒が設定されている
- readやwriteでエラーが発生した時は、接続(connectTCP)からやり直す必要がある
- TCP/IP機能を利用するためには、利用するSIMカードでパケット通信が利用できる必要がある。また、契約プランによっては通信料金が高額になる場合があるため注意すること
- 利用するSIMカードによっては、使用できるポート番号に制限(80番のみ等)がある場合がある。
- 1バイト単位でのread/writeは、実行効率が悪く、かつ実行速度が遅い。そのため、できるだけ複数バイト単位でread/write関数を呼び出して処理することが望ましい。

6. TCP/IP機能の関数　connectTCP

int connectTCP(const char* server, int port)	
機能概要	指定したサーバ、ポート番号へ接続して、TCPコネクションを確立する
引数	**server**：接続するサーバのホスト名またはIPアドレス **port**：接続するポート番号
戻り値	**0**：正常に接続できた時
	0以外：エラーが発生した時（戻り値はエラー番号を表す）
補足	本機能の処理には、状況によって1～30秒程度掛かる。 serverには、IPアドレス("x.x.x.x"形式)またはホスト名を指定することができる。

【使い方の例】

```
char *svr = "www.googl.co.jp";
if (a3gs.connectTCP(svr, 80) == 0) {
   // GET Request for google site
   a3gs.write("GET / HTTP/1.0$n");
   a3gs.write("HOST:");
   a3gs.write(svr);
   a3gs.write("$n$n");
   //  Get resopnse..
}
else {
   Serial.println("Error: can't connect");
}
```

6. TCP/IP機能の関数　disconnectTCP

int disconnectTCP(void)	
機能概要	接続しているTCPコネクションを切断する
引数	なし
戻り値	**0**：正常に切断できた時
	0以外：エラーが発生した時（戻り値はエラー番号を表す）
補足	本機能の処理には、状況によって1～数秒程度掛かる。 本ライブラリの終了関数endでは、TCPコネクションは自動的に切断しない。そのため、必要に応じて必ず本関数を呼び出してTCPコネクションを明示的に切断すること。

6. TCP/IP機能の関数　read

- バリエーション1

uint8_t read(void)	
機能概要	現在のTCPコネクションから１バイトのデータを読み出す
引数	なし
戻り値	**0以外**：読み出した1バイトのデータ
	0未満：エラーが発生した時（コネクションがcloseされた、エラーが発生した、タイムアウトした）
補足	本関数の処理には、状況によって1～30秒程度掛かる 本関数は同期処理であるため、コネクションから1バイトを読み出すまで、エラーが発生するまで、あるいはタイムアウトするまで呼び出し元に制御は戻らない。

```
char *svr = "www.googl.co.jp";
if (a3gs.connectTCP(svr, 80) == 0) {
   // GET Request for google site
   a3gs.write("GET / HTTP/1.0$n");
   a3gs.write("HOST:");
   a3gs.write(svr);
   a3gs.write("$n$n");
   handleResponse();
}
else {
   Serial.println("Error: can't connect");
}
```

```
void hadleResponse(void) {
  char c;
  while ((c = a3gs.read()) != eof) {
    // cを処理する
  }
}
```


TABrain a3gsa

6. TCP/IP機能の関数　read

● バリエーション2

int read(char* result, int resultlength)	
機能概要	現在のTCPコネクションから最大resultlengthバイトのデータを読み出す
引数	**result**：[OUT] 読み出したデータを格納するバッファアドレス **resultlength**：呼び出し側で確保したバッファのサイズ（バイトサイズ）
戻り値	**1以上**：正常に読み出した時（読み出したバイト数を返す）
	0未満：エラーが発生した時（コネクションがcloseされた、エラーが発生した、タイムアウトした）
補足	本関数の処理には、状況によって1～30秒程度掛かる。 本関数は同期処理であるため、コネクションからresultlengthバイトを読み出すまで、コネクションがcloseされるまで、エラーが発生するまで、あるいはタイムアウトするまで呼び出し元に制御は戻らない。

6．TCP/IP機能の関数　write

- バリエーション1

int write(uint8_t c)	
機能概要	現在のTCPコネクションへ１バイトのデータを書き出す
引数	**c**：書き出すデータ
戻り値	**1**：正常に書き出した時
	0未満：エラーが発生した時（コネクションがcloseされた、エラーが発生した、タイムアウトした）
補足	本関数の処理には、状況によって1～30秒程度掛かる 本関数は同期処理であるため、コネクションへデータを書き出すまで、コネクションがcloseされるまで、エラーが発生するまで、あるいはタイムアウトするまで呼び出し元に制御は戻らない。

6. TCP/IP機能の関数　write

● バリエーション2

int write(const char* str)	
機能概要	現在のTCPコネクションへ文字列データを書き出す
引数	**str**：書き出す文字列データ('¥0'で終端する文字配列)
戻り値	**1以上**：正常に書き出した時（書き出したバイト数を返す）
	0未満：エラーが発生した時（コネクションがcloseされた、エラーが発生した、タイムアウトした）
補足	本関数の処理には、状況によって1～30秒程度掛かる。 本関数は同期処理であるため、コネクションへデータを書き出すまで、コネクションがcloseされるまで、エラーが発生するまで、あるいはタイムアウトするまで呼び出し元に制御は戻らない。

【使い方の例】

```
char *svr = "www.googl.co.jp";
if (a3gs.connectTCP(svr, 80) == 0) {
  // GET Request for google site
  a3gs.write("GET / HTTP/1.0$n");
  a3gs.write("HOST:");
  a3gs.write(svr);
  a3gs.write("$n$n");
  //  Get resopnse..
}
```

6. TCP/IP機能の関数　write

● バリエーション3

int write(const uint8_t* buffer, size_t sz)	
機能概要	現在のTCPコネクションへ指定したバイト数のデータを書き出す
引数	**buffer**：書き出すデータ（バイト配列） **sz**：データのサイズ（バイト数）
戻り値	**1以上**：正常に書き出した時（書き出したバイト数を返す）
	0未満：エラーが発生した時（コネクションがcloseされた、エラーが発生した、タイムアウトした）
補足	バリエーション2ではデータの中にヌル文字('¥0')を取り扱うことができないが、本バリエーションではデータをサイズで管理するため、データの中にヌル文字を含めることができる。 本関数の処理には、状況によって1～30秒程度掛かる。 本関数は同期処理であるため、コネクションへデータを書き出すまで、コネクションがcloseされるまで、エラーが発生するまで、あるいはタイムアウトするまで呼び出し元に制御は戻らない。

【使い方の例】

```
char *svr = "192.168.1.1";
uint8_t  binaryData[] = { 0x0, 0x1, 0x2, 0x3, ..};
if (a3gs.connectTCP(svr, 8080) == 0) {
  a3gs.write(binaryData, sizeof(binaryData));
}
```

7．プロファイルの関数

提供関数ライブラリ

分類	メソッド名	機能概要	補足
プロファイル	setDefaultProfile	デフォルトプロファイルを設定	
	getDefaultProfile	デフォルトプロファイルを取得	

【注意事項】プロファイルは、通信サービス事業体が提供するSIMカードを利用するための設定情報である。
詳細は、setDefaultProfile 関数の説明を参照。

7. プロファイルの関数　setDefaultProfile

int setDefaultProfile(int profileNum)	
機能概要	デフォルトのプロファイルを指定した番号に設定する
引数	**profileNum**:デフォルトに指定するプロファイル番号 （1～**a3gsMAX_PROFILE_NUMBER**）
戻り値	**0**：正常に設定できた時
	0以外：設定できなかった時（引数の指定が間違っている等）
補足	設定したデフォルトのプロファイル番号は、IEM内のフラッシュROMに記録されるため、電源をOFFにしても維持される（再度、本ライブラリにて設定するまで有効である） IEM製品版3Gシールド(Ver 1.0)の出荷時の設定プロファイル情報は、下記の通りである： No.1　docomo の mopera.net サービス No.2　IIJ の iijmio サービス（＝デフォルトプロファイル）

7. プロファイルの関数　getDefaultProfile

int getDefaultProfile(int　*profileNum)	
機能概要	デフォルトのプロファイル番号を取得する
引数	**profileNum**:[OUT] 取得したデフォルトのプロファイル番号 (1～**a3gsMAX_PROFILE_NUMBER**)
戻り値	**0**：正常に取得できた時
	0以外：取得できなかった時
補足	

【使い方の例】

```
int pfNum;
if (a3gs.getDefaultProfile(&pfNum) == 0) {
  Serial.print("Default Profile No is ");
  switch (pfNum == 1)
    case 1 :
      Serial.println("docomo mopera");
    case 2 :
      Serial.println("IIJmio");
    default :
      Serial.println("unknown profile");
}
```

Ⅳ編　a3gs のインストール

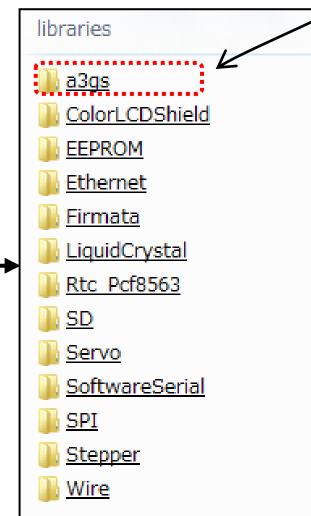
1．a3gs.zip ファイル

- ３Gアライアンスからは、ライブラリおよびサンプルスケッチ内容が、a3gs.zip（圧縮）ファイルとして提供されます。　（随時バージョンアップ版を提供して参ります）

名前	種類	サイズ	圧縮率
sample	ファイル フォルダー		
a3gs.cpp	CPP ファイル	35 KB	78%
a3gs.h	H ファイル	6 KB	64%
keywords	テキスト ドキュメント	2 KB	72%

- このzipファイルを解凍し、Arduino のIDE（統合化開発環境）のファイル群にある「libraries」の配下に複写（移動）をしてください。

Arduino IDE フォルダ	フォルダ	
最新版ArduinoIDE arduino-1.0.1-windows 配下	**drivers**	
	examples	
	hardware	
	java	
	lib	
	libraries	**※ こちらの配下に複写してください**
	reference	
	tools	



2．a3gs.zip ファイルの概要

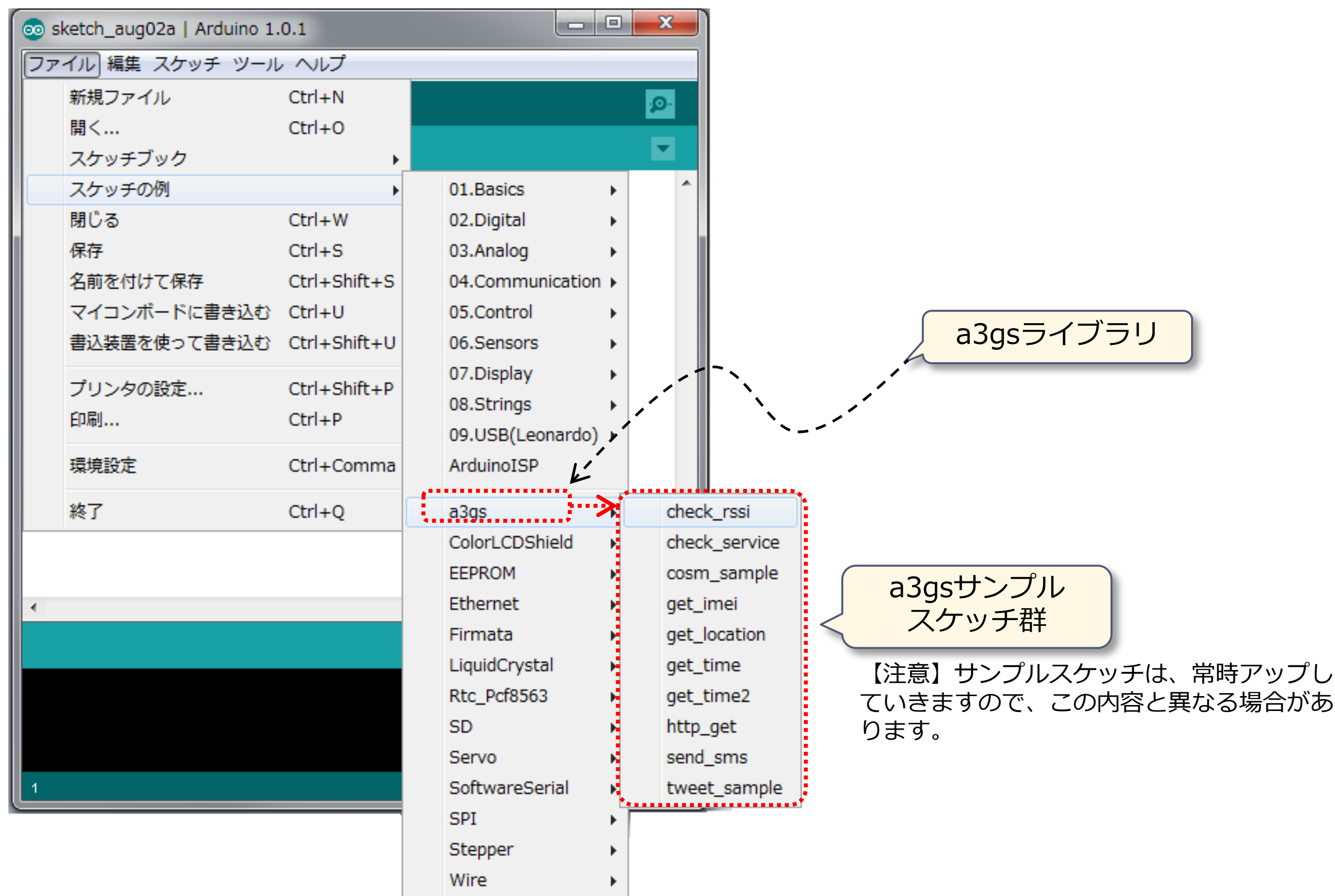
- 提供される a3gs.zip には、以下のファイル群が含まれています。鍵括弧内【ページ】は、V編のサンプル・スケッチのページです。

フォルダ/ファイル（スケッチ）		概　要
example【事例】	¥blink_led¥blink_led.ino【p.73】	LEDを点滅させるスケッチ※
	¥check_rssi¥check_rssi.ino【p.58】	電波状態を確認するスケッチ※
	¥check_service¥check_service.ino【p.59】	利用できるネットワークサービスを確認するスケッチ※
	¥cosm_sample¥cosm_sample.ino【p.60】	無償クラウドCOSMを利用したスケッチ
	¥get_imei¥get_imei.ino【p.63】	IEMモジュールのIDを取得するスケッチ※
	¥get_location¥get_location.ino【p.64】	現在位置を取得するスケッチ※
	¥get_status¥get_status.ino	getStatusを呼び出すスケッチ※
	¥get_time¥get_time.ino【p.65】	日付・日時を取り出すスケッチ※
	¥get_time2¥get_time2.ino【p.66】	'70年1月1日からの総合秒数を取り出すスケッチ※
	¥http_get¥http_get.ino【p.67】	httpGETを使うサンプルスケッチ
	¥on_sms¥on_sms.ino【p.74】	SMSの着信をチェックするスケッチ★
	¥sample_TCPIP¥sample_TCPIP.ino【p.75】	TCP/IP機能を利用するサンプルスケッチ
	¥send_sms¥send_sms.ino【p.68】	ショートメッセージを送信するスケッチ★
	¥set_baudrate¥set_baudrate.ino	通信速度を変更するスケッチ※ ＜注意＞
	¥set_defaultprofie¥set_defaultprofie.ino【p.78】	デフォルトのプロファイル(APN)を切り替えるスケッチ※
	¥tweet_sample¥tweet_sample.ino【p.69】	Tweetを活用したスケッチ
a3gs.cpp	"a3gs" 実装ファイル	
a3gs.h	"a3gs" ヘッダーファイル	
keywords.txt	a3gsライブラリのキーワード定義	

※印のついたスケッチは、適正以外のSIMカードでも関係なく起動。★印のついたスケッチは、音声通話用SIMカードで起動。
＜注意＞IEM通信モジュールとArduinoとの通信速度の設定で、内容理解の上利用。

3．a3gs インストール完了の確認

- a3gsのライブラリおよびサンプルスケッチが正しくインストールされた場合には、Arduino IDE環境下のプルダウンメニュー「ファイル（File)」の「スケッチの例」において、以下のように画面表示されます。



【注意】サンプルスケッチは、常時アップしていきますので、この内容と異なる場合があります。

Ⅴ編 サンプル・スケッチ

【注意事項】本稿で扱うスケッチ（プログラム）は、IDEバージョン Arduino 1.0.1上で作成・確認したものです。その他のバージョンでは稼働しない場合があります。

1． check_rssi.ino 【受信信号（電波）強度取得】

- Arduino上の３Gシールドが、３G電波を捉えているかを、数値によって返します。

【サンプルスケッチ（check_rssi.ino）】

```
// A3GS sample sketch.1  -- getRSSI

#include <SoftwareSerial.h>
#include "a3gs.h"

void setup()
{
  Serial.begin(9600);
  delay(3000);  // Wait for Start Serial Monitor
  Serial.println("Ready.");

  Serial.print("Initializing.. ");
  if (a3gs.start() == 0 && a3gs.begin() == 0) {
    Serial.println("Succeeded.");
    int rssi;
    if (a3gs.getRSSI(rssi) == 0)
      Serial.print("RSSI = ");
      Serial.print(rssi);
      Serial.println (" dBm");
  }
  else
    Serial.println("Failed.");
  Serial.println("Shutdown.. ");
  a3gs.end();
  a3gs.shutdown();
}

void loop()
{
}
```

電波強度を引数rssiに返す

シリアルモニタに
表示される内容（例）

```
Ready.
Initializing.. Succeeded.
RSSI = -75 dBm
Shutdown..
```

「-75」は電波強度

電波強度は必ずマイナス値で返却
「0」に近いほど電波強度は強い

2. check_service.ino 【SIMカードサービス形態取得】

- SIMカードのサービス形態を取得し表示するスケッチです。

【【サンプルスケッチ（check_sercvce.ino）】

```
// A3GS sample sketch.2  -- getServices
#include <SoftwareSerial.h>
#include "a3gs.h"

void setup()
{
  Serial.begin(9600);
  delay(3000);  // Wait for Start Serial Monitor
  Serial.println("Ready.");

  Serial.print("Initializing.. ");
```

```
  if (a3gs.start() == 0 && a3gs.begin() == 0) {
    Serial.println("Succeeded.");
    int rstatus;
    if (a3gs.getServices(status) == 0) {
      switch (status) {
        case a3gsSRV_NO :
          Serial.println("No Service.");     break;
        case a3gsSRV_PS :
          Serial.println("Packet Service Only.");      break;
        case a3gsSRV_CS :
          Serial.println("Voice Service Only."); break; // also SMS
        case a3gsSRV_BOTH :
          Serial.println("Packet And Voice Services."); break;
        default :     break;
      }
    }
  }
  else
    Serial.println("Failed.");

  Serial.println("Shutdown..");
  a3gs.end();
  a3gs.shutdown();
}

void loop() {}
```

【シリアルモニタ表示例】

```
Ready.
Initializing.. Succeeded.
Packet And Voice Services.
Shutdown..
```

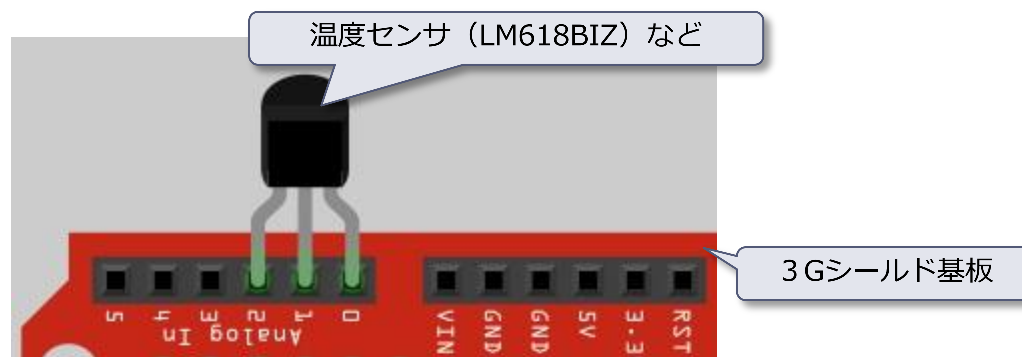


a3gsSRV_NO（＝0）：サービス利用不可
a3gsSRV_PS（＝1）：パケット通信サービスのみ
a3gsSRV_CS（＝2）：音声通信(+SMS)サービスのみ
a3gsSRV_BOTH（＝3）：パケット通信、音声通信(SMS)いずれも

3．cosm_sample.ino① 【クラウドとの連携】

- 無償（フリー）クラウドにセンサ値を送り、蓄積し、閲覧し、可視化するなどの共有機能を紹介します。センサネットワークの実現例として参考にしてください。
- 簡単に、早くクラウドへのデータ送信・蓄積が、実現することができます。
- あらかじめCOSMの登録（メールアドレス、ID、PW設定）およびID「**Feed ID**」/キー「**Key表示**」の取得を行っておいてください。登録およびIDとキーの取得は、【添付資料. 3】をご参考ください。

- ここでのサンプルスケッチでは、温度センサ（LM618BIZ）をA0(GND)、A1(Vout)、A2(Vs）に接続して利用しています。手持ちのセンサなどを使って、スケッチを変更して試してみてください。



3．cosm_sample.ino② 【クラウドとの連携】

【サンプルスケッチ（cosm_sample.ino）】

```
// A3GS sample  sketch.1  -- httpPOST (Use Cosm.com Colud Service)
#include <SoftwareSerial.h>
#include "a3gs.h"
#define LM61BIZ_Pin 1    // LM61BIZ output pin: A1

const char *server = "api.cosm.com";
const char *path = "v2/feeds/#####.csv?_method=put";
const char *header="X-ApiKey:***********$r$nContent-Type: text/csv$r$n";
int port = 80;

char res[30];
char body[20];
int len;
```

Feed IDが入る

Keyが入る

Feed IDとKeyは、予め
cosm.comに登録し、
取得しておく【添付資料.3】

```
int getTemp(void)
{
  int mV = analogRead(LM61BIZ_Pin) * 4.88;
  return (mV- 600);
}
```

センサ設定

```
void setup()
{
  pinMode(14, OUTPUT);  // A0(LM61BIZ - GND)
  digitalWrite(14, LOW);
  pinMode(16, OUTPUT);  // A2(LM61BIZ - VSS+)
  digitalWrite(16, HIGH);

  Serial.begin(9600);
  delay(3000);  // Wait for Start Serial Monitor
  Serial.println(">Ready.");

  Serial.print("Initializing.. ");
  if (a3gs.start() == 0 && a3gs.begin() == 0)
    Serial.println("Succeeded.");
  else {
    Serial.println("Failed.");
    while (1) ;  // STOP
  }
}
```

Feed IDとKeyは、予め
cosm.comに登録し、
取得しておく【添付資料.3】

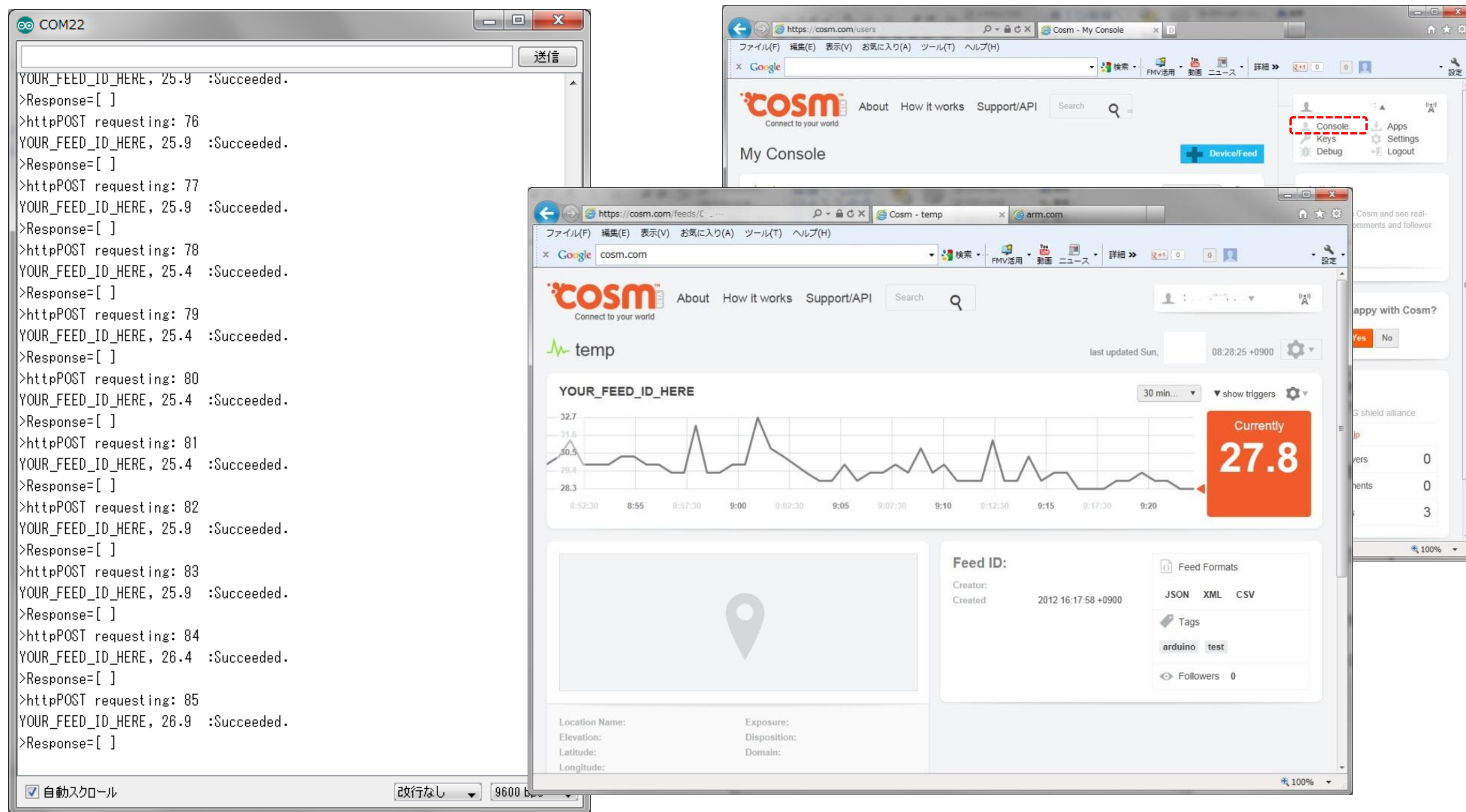
```
void loop()
{
  static int Count = 1;

  Serial.print(">httpPOST requesting: ");
  Serial.println(Count++, DEC);
  len = sizeof(res);
  int temp = getTemp();
  sprintf(body, "YOUR_FEED_ID_HERE,%d.%d", temp/10,temp%10);
  if (a3gs.httpPOST(server, port, path, header, body, res, &len) == 0) {
    Serial.println("Succeeded.");
    Serial.print(">Response=[");
    Serial.print(res);
    Serial.println("]");
  }
  else
    Serial.println("Failed.");
  delay(30000);  // take an interval
}
```

httpPOST利用

3．cosm_sample.ino③　【クラウドとの連携】

- cosm_sampleの実行結果事例



TABrain a3gsa

4． get_imei.ino 【IEMモジュール携帯端末識別番号取得】

- IEMモジュールの固有ID（IMEI：携帯端末識別番号）を取得し表示するスケッチです。

【サンプルスケッチ（get_time.ino）】

```
// A3GS sample sketch.4 -- getIMEI

#include <SoftwareSerial.h>
#include "a3gs.h"

void setup()
{
  Serial.begin(9600);
  delay(3000);  // Wait for Start Serial Monitor
  Serial.println("Ready.");

  Serial.print("Initializing.. ");
  if (a3gs.start() == 0 && a3gs.begin() == 0) {
    Serial.println("Succeeded.");
    char imei[a3gsIMEI_SIZE];
    if (a3gs.getIMEI(imei) == 0)
       Serial.print("IMEI: ") ;
       Serial.println(imei) ;
  }
  else
    Serial.println("Failed.");

  Serial.println("Shutdown..");
  a3gs.end();
  a3gs.shutdown();
}

void loop()  { }
```

IMEI取得関数

【シリアルモニタ表示例】

```
Ready.
Initializing.. Succeeded.
IMEI: 354563020267950
Shutdown..
```

IMEI
（携帯端末識別番号）

5. get_location.ino 【現在位置取得】

- 現在位置情報をGPSより取得表示するサンプルです。

```
// A3GS sample sketch.5  -- getLocation

#include <SoftwareSerial.h>
#include "a3gs.h"

void setup()
{
  Serial.begin(9600);
  delay(3000);    // Wait for start serial monitor
  Serial.println("Ready.");

  Serial.print("Initializing.. ");
  if (a3gs.start() == 0 && a3gs.begin() == 0) {
    Serial.println("Succeeded. It maybe takes several minutes.");
    char lat[15], lng[15];
    if (a3gs.getLocation(a3gsMPBASED, lat, lng) == 0) {
      Serial.print("OK: ");
      Serial.print(lat);
      Serial.print(", ");
      Serial.println(lng);
    }
    else
      Serial.println("I don't know this location.");
  }
  else
    Serial.println("Failed.");

  Serial.println("Shutdown..");
  a3gs.end();
  a3gs.shutdown();
}
void loop()  {}
```

位置情報取得

【注意事項】 GPS機能は、アンテナ特性が影響しますので、アンテナの感度の良いもので、なるべく周りの障害物を避け、さらに周辺にノイズ発生源が無いなどの条件で、位置情報が取得できます。

【シリアルモニタ表示例】

```
Ready.
Initializing.. Succeeded. It maybe takes several minutes.
OK:****** ,******
Shutdown..
```

緯度、経度を表示



TABrain a3gsa

6． get_time.ino 【時刻取得1】

- IEMモジュールにあるタイマー機能を使って、年月日および時刻を取得するスケッチです。

【サンプルスケッチ（get_time.ino）】

```
// A3GS sample  sketch.6  -- getTime

#include <SoftwareSerial.h>
#include "a3gs.h"

void setup()
{
  Serial.begin(9600);
  delay(3000);  // Wait for Start Serial Monitor
  Serial.println("Ready.");

  Serial.print("Initializing.. ");
  if (a3gs.start() == 0 && a3gs.begin() == 0) {
    Serial.println("Succeeded.");
    char date[a3gsDATE_SIZE], time[a3gsTIME_SIZE];
    if (a3gs.getTime(date, time) == 0) {
      Serial.print(date);
      Serial.print(" ");
      Serial.println(time);
    }
  }
  else
    Serial.println("Failed.");

  Serial.println("Shutdown..");
  a3gs.end();
  a3gs.shutdown();
}

void loop()  { }
```

時刻取得

【シリアルモニタ表示例】

```
Ready.
Initializing.. Succeeded.
2012/10/05 07:29:00
Shutdown..
```

【注意事項】 現在時刻の取得は、IEMモジュールの継続運用によって、設定されますので、初期状態では、1980年1月5日16:00の時刻か、日本時間の16時間遅れの時刻で表示される場合があります。利用時間経過とともに、日本時間に自動調整が行われます。

7． get_time2.ino 【時刻取得２】

- IEMモジュールにあるタイマー機能を使って、1970年1月1日からの総数秒を取得するスケッチです。

【サンプルスケッチ（get_time2.ino）】

```
// A3GS sample sketch.7  – getTime2

#include <SoftwareSerial.h>
#include "a3gs.h"

void setup()
{
  Serial.begin(9600);
  delay(3000);  // Wait for Start Serial Monitor
  Serial.println("Ready.");

  Serial.print("Initializing.. ");
  if (a3gs.start() == 0 && a3gs.begin() == 0) {
    Serial.println("Succeeded.");
    unit32_t  seconds;
    if (a3gs.getTime2(seconds) == 0) {
      Serial.print(seconds);
      Serial.println("  Sec.");
    }
  }
  else
    Serial.println("Failed.");

  Serial.println("Shutdown..");
  a3gs.end();
  a3gs.shutdown();
}

void loop()  { }
```

時刻取得

【シリアルモニタ表示例】

```
Ready.
Initializing.. Succeeded.
1344179843 Sec.
Shutdown..
```

【注意事項】 現在時刻の取得は、IEMモジュールの継続運用によって、設定されますので、初期状態では、1980年1月5日16:00での経過時間か、日本時間の16時間遅れの経過時間で表示される場合があります。利用時間経過とともに、日本時間の経過時間に自動調整が行われます。

TABrain a3gsa

8．http_get.ino 【サーバのレスポンス返却】

本スケッチでは、google サーバにアクセスし、レスポンスの格納先を表示させています。

【サンプルスケッチ（http_get.ino）】

```
// A3GS sample sketch.8  -- httpGET

#include <SoftwareSerial.h>
#include "a3gs.h"

const char *server = "www.arduino.cc";
const char *path = "";
int port = 80;

char res[a3gsMAX_RESULT_LENGTH+1];
int len;
```

Arduino サーバ

```
void setup()
{
  Serial.begin(9600);
  delay(3000);  // Wait for Start Serial Monitor
  Serial.println("Ready.");

  Serial.print ("Initializing.. ");
  if (a3gs.start() == 0 && a3gs.begin() == 0) {
    Serial.println("Succeeded.");
    Serial.print ("httpGET() requesting.. ");
    len = sizeof(res);
    if (a3gs.httpGET(server, port, path, res, &len) == 0) {
      Serial.println("OK!");
      Serial.print ("[");
      Serial.print (res);
      Serial.println("]");
    }
    else
      Serial.println("NG!");
  }
  else
    Serial.println("Failed.");

  Serial.println("Shutdown..");
  a3gs.end();
  a3gs.shutdown();

}

void loop() { }
```

【シリアルモニタ表示例】

```
Ready.
Initializing.. Succeeded.
httpGET() requesting.. OK!
[<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
  <title>Arduino - HomePage </title>
  <link rel="shortcut icon" type="image/x-icon"
href="http://arduino.cc/en/favicon]
Shutdown..
```

9． send_sms.ino 【ショートメッセージ送信】

- ショートメッセージを特定の携帯電話に送信するスケッチです。

```
// A3GS sample sketch.9 -- sendSMS (Japanease message version)
// PLEASE REPLACE "msn" WITH CORRECT TELEPHONE NUMBER BEFORE UPLOAD THIS SKETCH.

#include <SoftwareSerial.h>
#include "a3gs.h"

char *msn = "09012345678";   // Replace your phone number!
// char *msg = "TEST MESSAGE. HELLO!";  // ASCII String
char msg[] = { 0x53, 0x30, 0x8c,0x30, 0x6f, 0x30, 0xc6, 0x30, 0xb9, 0x30, 0xc8, 0x30, 0x67, 0x30, 0x59, 0x30, 0x00 };
  // Japanease written in UNICODE
```

SMSを受信する電話番号を設定

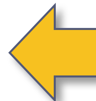
「これはテストです」（UNICODE）

【シリアルモニタ表示例】

```
Ready.
Initializing.. Succeeded.
SMS Sending.. OK!
Shutdown..
```

【モバイル側のSMS内容表示】

```
これはテストです
```



```
void setup()
{
  Serial.begin(9600);
  delay(3000);  // Wait for Start Serial Monitor
  Serial.println("Ready.");

  Serial.print("Initializing.. ");
  if (a3gs.start() == 0 && a3gs.begin() == 0) {
    Serial.println("Succeeded.");
    Serial.print("SMS Sending.. ");
    if (a3gs.sendSMS(msn, msg, a3gsCS_UNICODE) == 0)
      Serial.println("OK!");
    else
      Serial.println("Can't send SMS.");
  }
  else
    Serial.println("Failed.");

  Serial.println("Shutdown..");
  a3gs.end();
  a3gs.shutdown();
}
void loop()  {  }
```

【注意１】SMS機能は、音声通話ができるSIMカードでないと働きません。

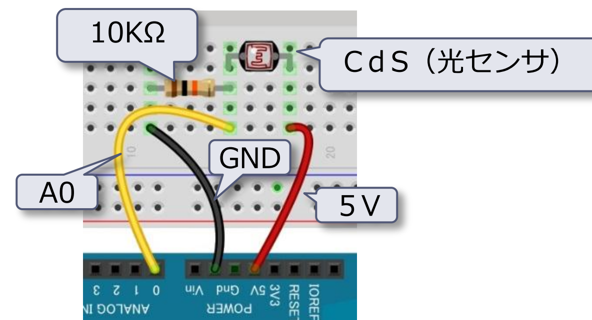
【注意２】送信できる文字コードは、アスキーコードか、漢字コード（日本語）ではUNICODEとなっています。

TABrain a3gsa

10. tweet_sample.ino① 【ツィートデータ送信】

- センサ値をtweetにつぶやきます。つまり３Gシールド上のセンサ値をTwitterに送ることができます。

※ センサ値は、光センサー（CdS）を利用



【サンプルスケッチ（tweet_sample.ino）】

```
// A3GS sample skech.10 – tweet_sample
#include <SoftwareSerial.h>
#include "a3gs.h"
const char *token ="136700536-*****************NxgiCMPGCqBXUEPl";
char *message = "３GShield Tweet Start" ;
  //-- Note: can't tweet same message continuously.
const int AnaogSensor_Pin = 0; // フォトICダイオード設定(アナログ０番)
```

Twtter トークン※

※ Twtterトークンは添付資料参照

【サンプルスケッチ（tweet_sample.ino）】

```
void setup()
{
 Serial.begin(9600);
  delay(3000);  // Wait for Start Serial Monitor
  Serial.println("Ready.");

  Serial.print("Initializing.. ");
  if (a3gs.start() == 0 && a3gs.begin() == 0) {
       Serial.println("Succeeded.");
       Serial.print("tweet() requesting.. ");
    if (a3gs.tweet(token, message) == 0)
      Serial.println("OK!");
    else
      Serial.println("NG!");
  }
  else
      Serial.println("Failed.");
}
```

```
void loop()
{   static uint16_t  prevLx = 0;
    uint16_t  lx = analogRead(AnaogSensor_Pin);
    Serial.print(" lx = "); Serial.print(lx);

    if (lx != prevLx) {
      sprintf(message, "3GShield Sensor Value = %d", lx);
      prevLx = lx;
      Serial.print("tweet() requesting[");
      Serial.print(lx);
      Serial.print("]..");
      if (a3gs.tweet(token, message) == 0)
        Serial.println("OK!");
      else
        Serial.println("NG!");
    }
    delay(30000);// 30秒ごとに出力
}
```

センサー値

10. tweet_sample.ino② 【ツィートデータ送信】

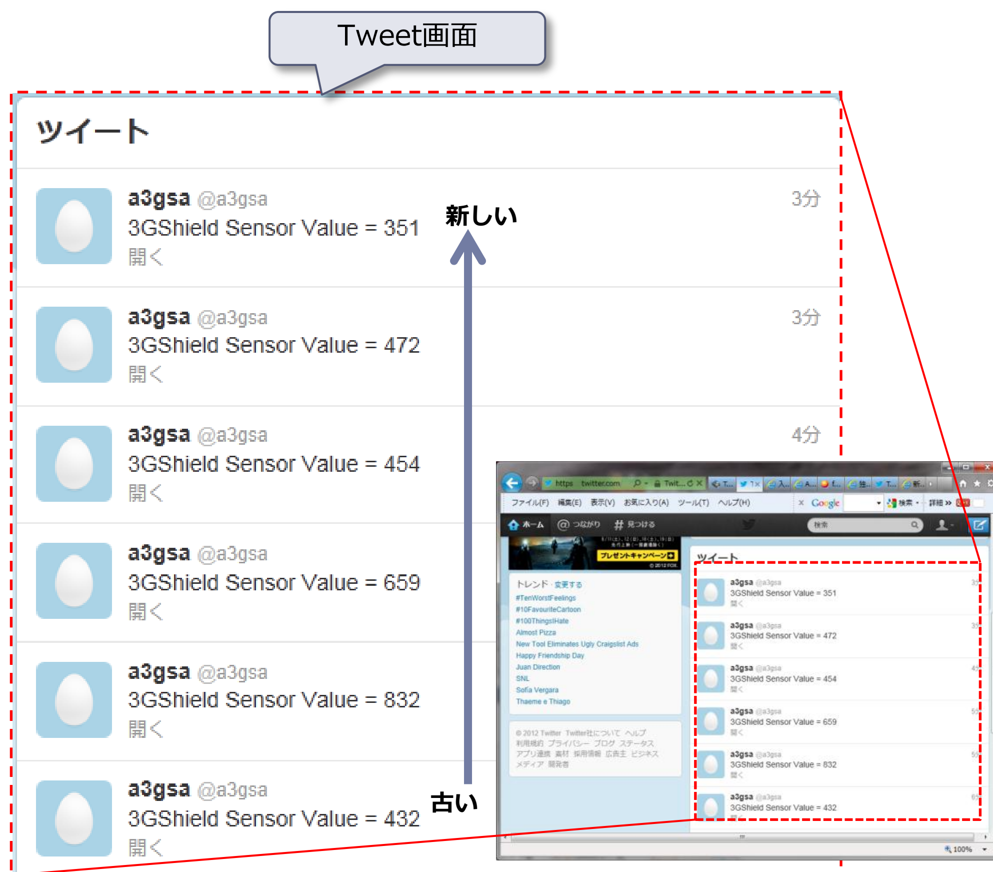
- 30秒ごとのセンサ値は、「ツイート」に送信され、またシリアル画面にも表示されます。

※ 以下の事例でのセンサ値は、光センサー（CdS）を利用

シリアル画面表示

```
COM6
送信
Ready.
Initializing.. Successeded.
tweet() requesting.. NG!
 lx = 432tweet() requesting[432]..OK!
 lx = 428tweet() requesting[428]..NG!
 lx = 832tweet() requesting[832]..OK!
 lx = 659tweet() requesting[659]..OK!
 lx = 454tweet() requesting[454]..OK!
 lx = 472tweet() requesting[472]..OK!
 lx = 351tweet() requesting[351]..OK!
自動スクロール　改行なし　9600 bps
```

古い ↓ 新しい

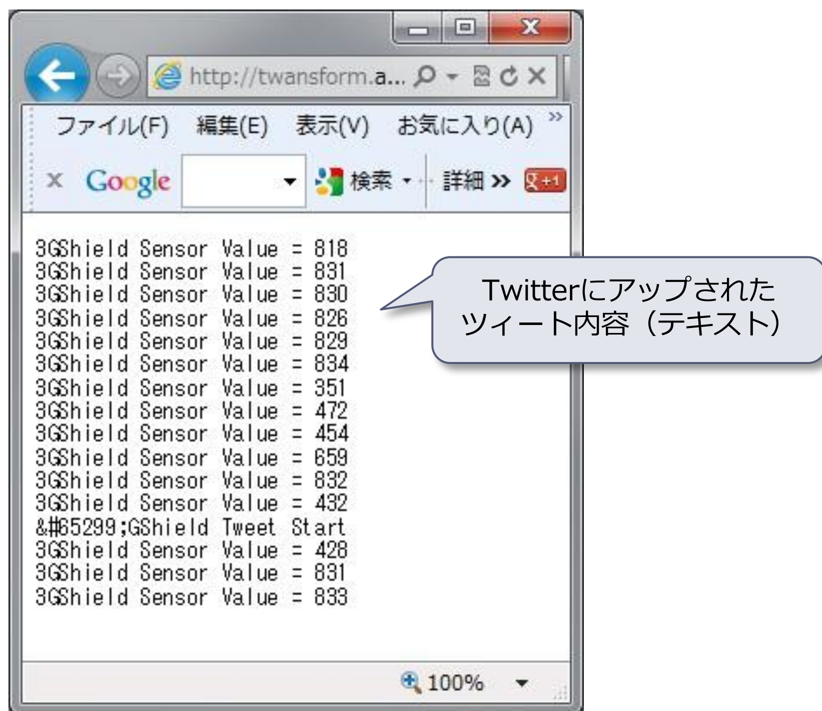


11. tweet_sample2.ino【ツィートデータ受信】

- 次にtweet_sampleでツイートにセンサ値をアップした内容を、受信するものに挑戦しましょう。
- tweetされたデータは、以下のようにして取り出しが可能です。

  http://arduino-tweet.appspot.com/******/text/1

  「******」には、Twitterのアカウント名が入ります。
- ここでは、サンプルのスケッチはありませんので、ユーザが独自に挑戦してみてください。



12. set_defaultprofile.ino 【プロファイル変更】

- 本スケッチは、SIMカードのプロファイルを書き替えるもので、NTTドコモ通信サービス（＝１）とIIJ個人向け通信サービス（IIJmio＝２）の切替を行う事例となります。

```
// A3GS sample skech.12 -- setDefaultProfile and getDefaultProfile

#include <SoftwareSerial.h>
#include "a3gs.h"
```

```
void setup()
{
  Serial.begin(9600);
  delay(3000);  // Wait for Start Serial Monitor
  Serial.println("Ready.");

  Serial.print("Initializing.. ");
  if (a3gs.start() == 0 && a3gs.begin() == 0) {
    Serial.println("Succeeded.");
    int no;
    if (a3gs.getDefaultProfile(&no) == 0) {
      Serial.print("Default Profile Number is ");
      Serial.println(no);
    }
    if (no == 1)
      no = 2;
    else
      no = 1;
    if (a3gs.setDefaultProfile(no) == 0) {
      Serial.print("Set Default Profile Number as ");
      Serial.println(no);
    }
    else
      Serial.println("Failed.");
  }
  else
    Serial.println("Failed.");
}

void loop()  {}
```

プロファイル取り出し

- このサンプルスケッチは、トグル（交互）設定機能となっています。

プロファイル切替え

【注意】SIMカードの提供は、通信キャリア以外にも、MVNOなどによっても提供されています。このSIMカードの種類は、現在多く出回っていて、この３Gシールドで使えるSIMカードは、IIJの法人向けSIMカードやIIJmio個人向けSIMカードなどに限定しています。
　必ずしもNTTドコモのSIMカードでも、一部の機能では稼働するとは限りません。
　（３Gシールドアライアンスでは、IIJmioの販売提供を行っていますので、別途ご連絡頂けますと、ご提供して参ります）

13. blink_led.ino 【LEDの点滅】

- 本スケッチは、3Gシールド基板上に搭載されているLED1のON/OFFを行う事例です。

```
// A3GS sample skech.13 -- setLED

#include <SoftwareSerial.h>
#include "a3gs.h"
```

```
#define  INTERVAL  1000  // Blink interval

void setup()
{
  Serial.begin(9600);
  delay(3000);  // Wait for Start Serial Monitor
  Serial.println("Ready.");

 _retry:
  Serial.print("Initializing.. ");
  if (a3gs.start() == 0 && a3gs.begin() == 0)
    Serial.println("Succeeded.");
  else {
    Serial.println("Failed.");
    Serial.println("Shutdown..");
    a3gs.end();
    a3gs.shutdown();
    delay(30000);
    goto _retry;  // Repeat
  }
  Serial.println("Blinking..");
}

void loop()
{
  a3gs.setLED(true);
  delay(INTERVAL);
  a3gs.setLED(false);
  delay(INTERVAL);
}
```

初期化失敗時は、リトライする

LEDのOn/Off処理

14. on_sms.ino 【SMSの着信時処理】

- 本スケッチは、SMSが着信した時に処理を行う事例です。
  ハンドラ関数lenOn内では最低限の処理を行い、それ以外の長い処理は
  フラグreceivedを立てることにより、loop()内で実行していることに注
  意してください。

```
// A3GS sample skech.14 -- onSMSReceived

#include <SoftwareSerial.h>
#include "a3gs.h"

const int ledPin = 13;
volatile boolean received = false;

char msg[a3gsMAX_SMS_LENGTH+1],
msn[a3gsMAX_MSN_LENGTH+1];

void ledOn(void)
{
  digitalWrite(ledPin, HIGH);  // LED on
  received = true;
}
```

SMS着信時に呼び出
されるハンドラ関数

```
void setup()
{
  Serial.begin(9600);
  delay(3000);  // Wait for Start Serial Monitor
  Serial.println("Ready.");
  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, LOW);  // LED off
  Serial.print("Initializing.. ");
  if (a3gs.start() == 0 && a3gs.begin() == 0) {
    Serial.println("Succeeded.");
    a3gs.onSMSReceived(ledOn);
  }
  else
    Serial.println("Failed.");
}

void loop()
{
  if (received) {
    Serial.println("SMS was received.");
    if (a3gs.readSMS(msg, sizeof(msg), msn, sizeof(msn)) == 0) {
      Serial.print("MSN: ");
      Serial.println(msn);
      Serial.print("SMS: ");
      Serial.println(msg);
    }
    digitalWrite(ledPin, LOW);  // LED off
    received = false;
    a3gs.onSMSReceived(ledOn);  // Re-set handler
  }
}
```

SMS着信時の処理

【シリアルモニタ表示例】

```
Ready.
Initializing.. Succeeded.
SMS was received.
MSN: 09012345678
SMS: 3g shield
```

送信側電話番号

送られてきたSMS
（最大約100文字程度）

TABrain a3gsa

15. sample_TCPIP.ino ① 【TCP/IP機能のサンプル】

- 本スケッチは、TCP/IP機能を使ってhttp通信を行う事例です。この例では、Arduinoの公式ページから、titleタグの中身を抜き出して、シリアルへ出力します。
改行文字等は、$文字を使ったエスケープシーケンスを使って記述します。

```
// A3GS sample skech.15-- connectTCP/disconnectTCP/read
//   A title is extracted from a homepage.

#include <SoftwareSerial.h>
#include <a3gs.h>

const char *server = "www.arduino.cc";  // URL to extract a title
const int port = 80;
char res[200];

boolean getTitle(char *res);

void setup()
{
  Serial.begin(9600);
  delay(3000);  // Wait for Start Serial Monitor
  Serial.println("Ready.");

 _redo:
  Serial.print("Initializing.. ");
  if (a3gs.start() == 0 && a3gs.begin() == 0) {
    Serial.println("Succeeded.");
    if (a3gs.connectTCP(server, port) != 0) {
      Serial.println("connectTCP() can't connect");
      goto _end;
    }
```

TCP/IPの接続処理

```
    // Send GET request
    a3gs.write("GET / HTTP/1.0$n");
    a3gs.write("HOST:");
    a3gs.write(server);
    a3gs.write("$n$n");
    // Receive response
    do {
      int nbytes;
      if ((nbytes = a3gs.read(res, sizeof(res)-1)) < 0) {
        Serial.println("read() failed");
        break;
      }
      res[nbytes] = '¥0';
    } while (! getTitle(res));
  }
  else
    Serial.println("Failed.");

  Serial.println("Shutdown..");
  a3gs.end();
  a3gs.shutdown();

  delay(15000);
  goto _redo;  // Repeat

 _end:
  while (1) ;
}

void loop() { }
```

HTTP/GETリクエスト

Titleタグが取得できるまで、レスポンスを読み続ける

15. sample_TCPIP.ino ② 【TCP/IP機能のサンプル】

```
boolean getTitle(char *p)
{
  char  *title;

  while (*p != '¥0') {
    if (*p++ != '<')
      continue;    // skip not tag
    if (strncmp((const char *)p, "title>", 6))
      continue;    // skip not title tag
    // title tag found
    p += 6;
    title = p;
    while (*p != '¥0' && *p != '<')
      p++;
    *p = '¥0';
    Serial.print(server);
    Serial.print(" : Page title is ¥"");
    Serial.print(title);
    Serial.println("¥"");
    return true;
  }
  return false;
}
```

レスポンスの中から<title>タグを見つける

<title>タグの中身だけを抽出する

【シリアルモニタ表示例】

```
Ready.
Initializing.. Succeeded.
www.arduino.cc : Page title is "Arduino -
HomePage "
Shutdown..
```

添付資料

【添付資料.1】a3gs.h ライブラリー覧表

※ Arduino GSM/GPRSシールド用ライブラリと互換性がある関数

分類	メソッド名	機能概要	補足
コントロール関係	getStatus※	3Gシールドの状態取得	
	begin※	ライブラリの初期化	
	end※	ライブラリの終了	
	restart※	3Gシールドのリセット	
	start※	3Gシールドの電源ON	
	shutdown※	3Gシールドの電源OFF	
	getIMEI	IMEIの取得	携帯端末固有番号
	setBaudrate	通信速度の設定	
	setLED1	LED1のON/OFF	
ショートメッセージ関係（SMS）	sendSMS※	SMSの送信	
	availableSMS※	SMSの受信状態チェック	
	readSMS※	SMSの読出し	
	onSMSReceived	SMS着信時のコールバック設定	
インターネット関係（Web）	httpGET※	GETメソッドの要求	
	httpPOST	POSTメソッドの要求	
	tweet※	Twitterへの投稿	*
インターネット関係（TCP）	connectTCP※	TCPコネクションを接続する	
	disconnectTCP※	TCPコネクションを切断する	
	read※	データを読み込む	
	write※	データを書き出す	
位置情報取得（GPS）関係	getLocation	現在位置の取得	内蔵GPSを使用
その他ライブラリ	getServices	利用可能サービスの取得	
	getRSSI	電波強度の取得	
	getTime	現在時刻の取得	日付・時刻形式
	getTime2	現在時刻の取得	通算秒形式
	getVersion	3Gシールド(gw3gアプリ)のバージョンの取得	

* 無償サービス「http://arduino-tweet.appspot.com/」を利用（要Twitterの登録）

【添付資料.2】tweetトークン取得

- 以下のサイトからトークン取得を行う。

  http://arduino-tweet.appspot.com/

Tweet Library for Arduino

Post messages to Twitter

How to begin:

Step 1 : Get a token to post a message using OAuth.

Step 2 : Add some Libraries to your Arduino IDE.

Step 3 : Run a sample sketch to tweet!

Notice

- The library uses this site as a proxy server for OAuth
- Please avoid sending more than 1 request per minute
- Twitter seems to reject repeated tweets with the same co

Reference

See Arduino: Playground

Authorize Arduino to use your account?

この連携アプリを認証すると、次の動作が許可されます。

- タイムラインのツイートを見る。
- フォローしている人を見る、新しくフォローする
- プロフィールを更新する。
- ツイートする

ユーザー名、またはメールアドレス

パスワード

保存する · パスワードを忘れた場合はこちら

連携アプリを認証　許可を取り消す

この連携アプリを認証しても、次の動作は許可されません。

- ダイレクトメッセージを見る。
- Twitterのパスワードを見る。

Your token is:

136700536-mDsv6n3pJO … xgiCMPGCqBXUEPl

登録しているID/PW を入れて承認

Twitterに登録していない場合は、別途Twitterの登録が必要です。

トークンが表示される

この「トークン」表示を利用

TABrain a3gsa

【添付資料.3】 COSMの利用登録①

- 以下のサイトからCOSMの利用登録を行い、
  cosm.com にアクセス

【添付資料.3 】 COSMの利用登録②

次の手順でデータ登録を行う。

Arduino選択

What type of device/feed do you want to add?

Arduino
Arduino Ethernet or using Ethernet Shield

Current Cost NetSmart

Twitter Stats
Create a test feed by adding your Twitter stats

Something Else

【登録】
① タイトル
② タグ（見出し）
を入力し
「Create」ボタンを押す

Configure your **Arduino**.

Step 1 - Title

Step 2 - Tags

Step 3 - Create

Create your new feed

Create

Success

Your new feed ID is **70094**. Paste this code into a new sketch to get started. See this tutorial for more details.

Arduino Sketch

```
/*
 Cosm sensor client

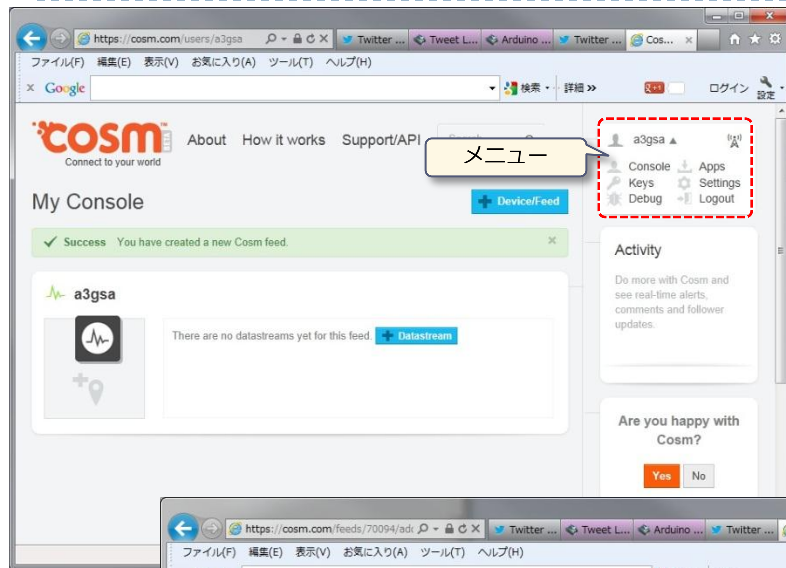
 This sketch connects an analog sensor to Cosm (http://www.cosm.com)
 using a Wiznet Ethernet shield. You can use the Arduino Ethernet shield, or
 the Adafruit Ethernet shield, either one will work, as long as it's got
 a Wiznet Ethernet module on board.

 This example has been updated to use version 2.0 of the Cosm.com API.
 To make it work, create a feed with a datastream, and give it the ID
 sensor1. Or change the code below to match your feed.
```

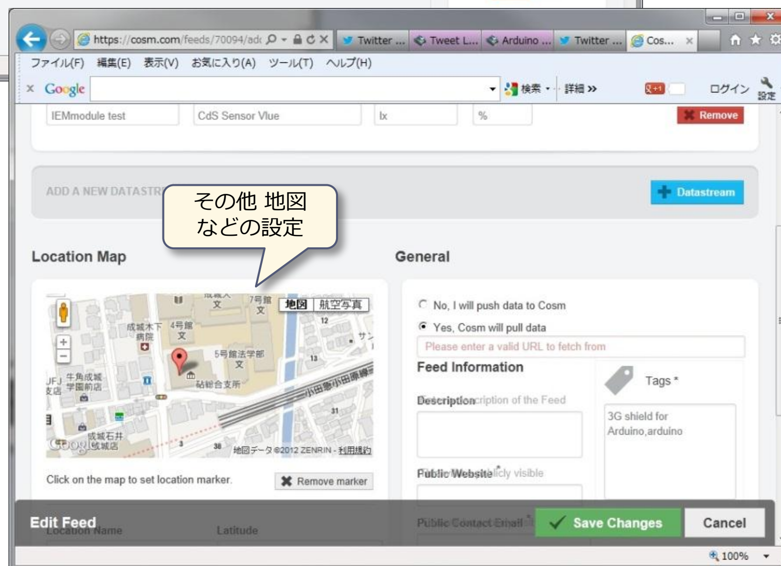
成功したら終了へ
→次ページ

Finish

【添付資料.3 】 COSMの利用登録③









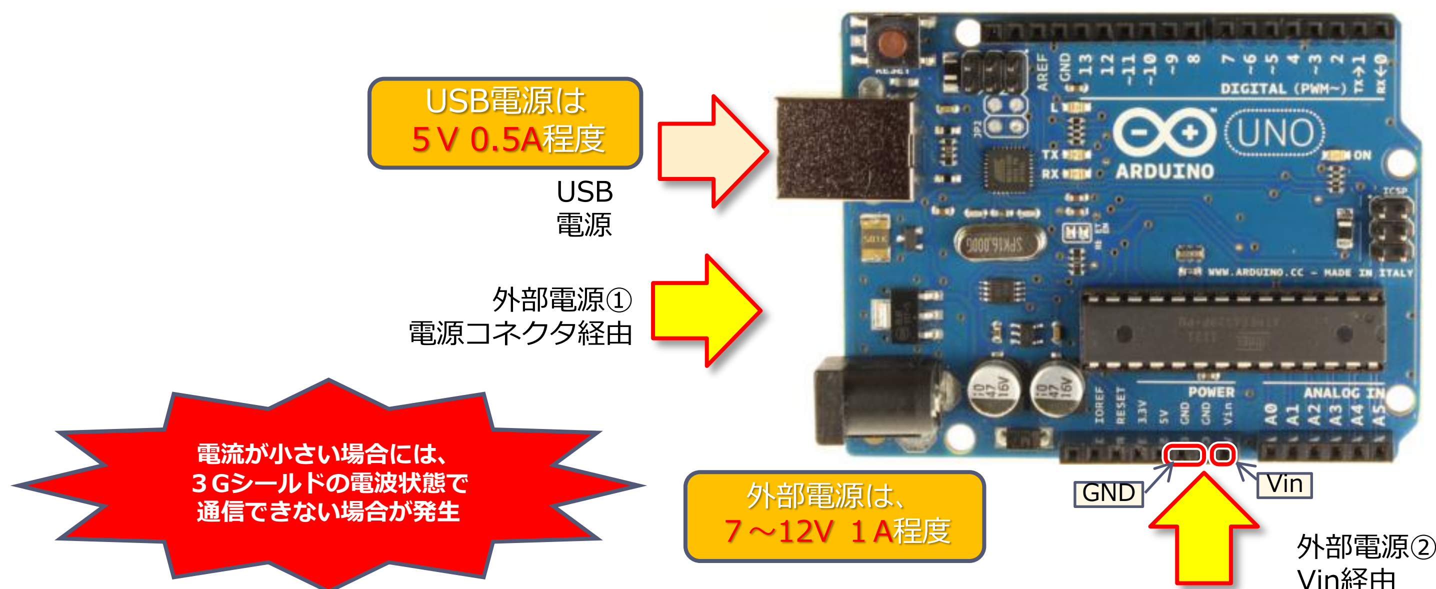
TABrain a3gsa

【添付資料.4 】語彙説明

- シールドとは、Arduino本体に装着できる拡張ボードのことを指す。
- スケッチとは、Arduino上で稼働するプログラムのこと。
- プロファイルとは、SIMカードを認識するためのもの（APN）で、予めIEMモジュールに、どのSIMカードで利用するかを設定する必要がある。（省略時設定は、別途記載している通り）
- a3gsとは、arduino 3G shield の略で、Arduino上で3gシールドを利用するためのライブラリ群となる。
- APN（プロファイル）とは、Access Point Name（アクセス・ポイント・ネーム）の略で、携帯電話ネットワーク上のデータ通信で必要となる接続先を指定する文字列となる。通信キャリアごとやMNVO（仮想移動体通信事業者）ごとに固有の名前が付けられている。
- GPSとは、Global Positioning Systemの略で、高度約2万Kmの複数のGPS衛星から電波を受信し、緯度と経度を割り出すシステム。
- gw3gとは、GateWay to 3Gの略で、IEMモジュールに組み込んだBrewMPのアプリケーション群。
- IDEとは、統合的な開発環境のことを指す。
- IEMとは、Internet of Everything Moduleの略で、クアルコム設計の通信モジュールの概念となる。
- IMEIとは、International Mobile Subscriber Identityの略で、携帯端末の識別番号で、IEM製品版のIMEIは３G通信モジュール（IEM）の固有識別番号となる。
- MVNO（仮想移動通信事業者）とは、Mobile Virtual Network Operatorの略で、携帯電話やPHSなどの物理的な移動体回線網を自社で持つことなく、通信キャリアから借りて、自社ブランドで通信サービスを行う事業者のこと。
- RSSI値とは、Received Signal Strength Indicatorの略で、「受信信号強度」のことを意味し、受信機入力に入る受信信号の強度を示す数値となる。
- UNICODEは、世界的な標準の文字コードのひとつである。
- UTF-8は、UCSとUNICODEで使える8ビット符号単位の文字符号化形式及び文字符号化スキーム。

【添付資料.5 】電源供給について

- ３Gシールドは、これまでの調査で、一時的に約200mA~600mAの電力を消費することが分かりました。
- このことで、電波状態やセンサ類の利用などの状況によっては、PCによるUSB電源（５V500mA）では、電源供給できない状態になり、通信できない場合が発生します。この障害の場合には、外部電源（推奨は９V1.3A）を別途、取って頂くことで、通信が可能となります。
- この他、USB接続ではなく、外部電源のみで利用される場合には、消費電力（特に電流）をチェック（測定）して、安定したバッテリー電力を使ってご使用ください。（12年12月現在３Gシールドアライアンスでは、消費電力の測定中で、一部Facebookなどで公開していきます。またアライアンス・メンバーには情報提示を行っていきます）
- 二次電池（ニッケル水素やリチウムイオン）などのバッテリーをご利用される場合には、電池の電圧・電流（その積による電力）能力を十分満たすものを外部電源としてご利用ください。外部電源は、コネクタ部分からと、VinとGNDの２つのピン接続のいずれかで取ることができます。



TABrain a3gsa